```
# -*- coding: utf-8 -*-
"""
/***************************************************************************
 ParteObraDockWidget
                                 A QGIS plugin
 Parte de Obra con Fotos de Campo Comentadas incluso Incidencias
 Generated by Plugin Builder: http://g-sherman.github.io/Qgis-Plugin-Builder/
                             -------------------
        begin                : 2023-05-18
        git sha              : $Format:%H$
        copyright            : (C) 2023 by Antonio Carlos Benavides García
        email                : antonio.benavides@usal.es
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""

import os
# para ruta y para borrar el archivo existente previo
from os import remove
import pathlib # importa la libreria de rutas pathlib y va así a pelo
from pathlib import Path

import qgis.gui
from PyQt5.QtGui import QPixmap
from qgis.PyQt.QtWidgets import QAction, QFileDialog, QMessageBox, QInputDialog, QLabel, QDialog
from qgis.gui import QgsFileWidget, QgsMessageBar
from qgis.PyQt import QtGui, QtWidgets, uic
from qgis.PyQt.QtCore import pyqtSignal
from qgis.core import QgsProject,QgsDataProvider,QgsVectorLayer,QgsFeature,QgsGeometry,QgsField,QgsPointXY, QgsVectorFileWriter, QgsRectangle, QgsRasterLayer
# QgsVectorFileWriter esta clase es para escribir en disco
from PyQt5.QtCore import QDate, QDateTime

from qgis import processing #necesario para ejecutar comandos de la caja de herramientas


FORM_CLASS, _ = uic.loadUiType(os.path.join(
    os.path.dirname(__file__), 'ParteObra_dockwidget_base.ui'))


class ParteObraDockWidget(QtWidgets.QDockWidget, FORM_CLASS):

    closingPlugin = pyqtSignal()

    def __init__(self, parent=None):
        #    C O N S T R U C T O R
        #        L L A M A D A  A  F U N C I O N E S

        """Constructor."""
        super(ParteObraDockWidget, self).__init__(parent)
        self.setupUi(self)

        #**** PESTAÑA 1 ******
        self.qfw_selector.setStorageMode(QgsFileWidget.GetDirectory)
        self.pb_cargar_capa.clicked.connect(self.cargar)
        self.btn_importar.clicked.connect(self.importar)
        #desde el botón btn_importar se llama a la función importar
        self.tw_seleccion.itemSelectionChanged.connect(self.focoimagen)


        #**** PESTAÑA 2 ******
        self.btn_actualizar.clicked.connect(self.actualizar_capas)
```

```
70            global lista_capas #variable global
71            capas = [capa for capa in QgsProject.instance().mapLayers().values() if capa.
              type() == QgsVectorLayer.VectorLayer]
72            #capas=QgsProject.instance().mapLayers().values() # coge los valores de las
              capas de la instancia actual del proyecto y lo guarda
73            lista_capas=[]
74            # para una variable una_capa en la lista de capas
75            # recorre todas las capas y para cada una de ellas te quedas con el nombre en
              una lista
76            lista_capas=[una_capa.name() for una_capa in capas]
77            self.cmb_capas.clear() #borra el contenido del cuadro combinado
78            self.cmb_capas.addItems(lista_capas)
79            self.cmb_capas.currentIndexChanged.connect(self.elegir) #al seleccionar
              desplegable activa la función elegir
80            self.pb_registra.clicked.connect(self.registra_cambios)
81            self.btn_pnoa.clicked.connect(self.pnoa)
82
83        def pnoa(self,event):
84            #    C A R G A   R A S T E R   O R T O   P N O A
85
86            #pnoa max actualidad. Carga el ráster del servicio WMS
87            urlWithParams =
              "crs=EPSG:4326&format=image/png&layers=OI.OrthoimageCoverage&styles&url=http:/
              /www.ign.es/wms-inspire/pnoa-ma"
88            wmsLayer3 = QgsRasterLayer(urlWithParams, 'PNOA MA', 'wms')
89            QgsProject.instance().addMapLayer(wmsLayer3)
90
91        def closeEvent(self, event):
92            self.closingPlugin.emit()
93            event.accept()
94
95        def actualizar_capas(self, event):
96            #    A C T U A L I Z A   L I S T A   C A P A S
97
98            #Actualiza todas las capas del proyecto de tipo vectorial y las incorpora en
              una lista de la que seleccionamos una
99            global lista_capas #variable global
100           capas = [capa for capa in QgsProject.instance().mapLayers().values() if capa.
              type() == QgsVectorLayer.VectorLayer]
101           #capas=QgsProject.instance().mapLayers().values() # coge los valores de las
              capas de la instancia actual del proyecto y lo guarda
102           lista_capas=[]
103           # para una variable una_capa en la lista de capas
104           # recorre todas las capas y para cada una de ellas te quedas con el nombre en
              una lista
105           lista_capas=[una_capa.name() for una_capa in capas]
106           self.cmb_capas.clear() #borra el contenido del cuadro combinado
107           self.cmb_capas.addItems(lista_capas)
108
109       def elegir(self, event):
110           #    C A P A   A C T I V A
111
112           #pondrá en el label el elemento seleccionado en el combo
113           self.txt_activa.setText(self.cmb_capas.currentText())
114
115       def importar(self, event):
116           #    I M P O R T A C I O N   F O T O S
117           #         I N T E R S E C C I Ó N
118           #              C A R G A   L A   C A P A
119
120           #Importa fotos geoetiquetadas y luego las intersecta con trazabilidad y carga
              la capa
121           self.lb_casco.setText("")
122           self.lb_casco2.setText("")
123
124           #llamada a importar fotos caja herramientas
125
126           ruta0=self.qfw_traza.filePath()
127           ruta=self.qfw_selector.filePath()
128           ruta2=ruta + '/previo.gpkg'
129
130           path=pathlib.PurePath(ruta) #ruta de la carpeta
```

```
131             ultima_carpeta=path.name #nombre de la carpeta ultima
132             ruta3=ruta+'/'+ultima_carpeta+'.gpkg' #nombre del archivo tomado de la
                carpeta.gpkg
133
134             # si ya existía, advierte y se detiene, sino existe prosigue
135             if os.path.exists(ruta3):
136                 self.lb_casco.setText("Atención: El archivo existe")
137                 self.lb_casco2.setText("Bórralo y carga de nuevo")
138                 return
139
140             processing.run("native:importphotos", {'FOLDER': ruta,
141                 'RECURSIVE':True,
142                 'OUTPUT':ruta2,
143                 'INVALID':''})
144
145             #intersección con trazabilidad
146
147             processing.run("native:intersection", {'INPUT': ruta2,
148                 'OVERLAY':ruta0,
149                 'INPUT_FIELDS':'',
150                 'OVERLAY_FIELDS':'',
151                 'OVERLAY_FIELDS_PREFIX':'',
152                 'OUTPUT':ruta3})
153                 # la salida se llama como la carpeta
154
155
156             #carga la capa intersectada ; añade la capa ultima_carpeta
157             vlayer = QgsVectorLayer(ruta3, ultima_carpeta)
158             vlayer.updateExtents()
159             QgsProject.instance().addMapLayer(vlayer)
160
161         def cargar(self,event):
162             #     C A R G A   L A   T A B L A
163
164             #de la lista de capas elige una y se selecciona en el tw_seleccion de
                QtDesigner
165             capas_seleccionadas=QgsProject.instance().mapLayersByName(self.txt_activa.text
                ())
166             capa_seleccionada=capas_seleccionadas[0]
167             self.tw_seleccion.setRowCount(0)
168             registros=capa_seleccionada.getFeatures() #de la capa seleccionada activa sus
                campos
169             for registro in registros: #carga el contenido de los campos de la fila en la
                tabla Table Widget de QtDesigner
170                 self.tw_seleccion.insertRow(self.tw_seleccion.rowCount())
171                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,0,QtWidgets.
                    QTableWidgetItem(str(registro["incidencia"])))
172                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,1,QtWidgets.
                    QTableWidgetItem(registro["timestamp"].toString("dd/MM/yyyy")))
173                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,2,QtWidgets.
                    QTableWidgetItem(str(registro["DESCRIPCION"])))
174                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,3,QtWidgets.
                    QTableWidgetItem(str(registro["DESCRIPCION_TRAZA"])))
175                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,4,QtWidgets.
                    QTableWidgetItem(str(registro["CODIGO"])))
176                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,5,QtWidgets.
                    QTableWidgetItem(str(registro["longitude"])))
177                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,6,QtWidgets.
                    QTableWidgetItem(str(registro["latitude"])))
178                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,7,QtWidgets.
                    QTableWidgetItem(str(registro["photo"])))
179                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,8,QtWidgets.
                    QTableWidgetItem(str(registro["fid"])))
180
181         def focoimagen(self):
182             #    C E N T R A R   M A P A
183
184             fila=self.tw_seleccion.currentRow()
185
186             # Definir las coordenadas específicas (en este caso, latitud y longitud)
187             latitud = float(self.tw_seleccion.item(fila,6).text())
188             longitud = float(self.tw_seleccion.item(fila,5).text())
```

```
189             # Crear un objeto QgsRectangle con las coordenadas
190             map_pos = QgsPointXY(longitud, latitud) #referencia a la posicion del punto xy
191             rect = QgsRectangle(map_pos, map_pos) #este rectangulo, no tiene dimension es
                de 0x0 pero centraliza el punto
192             mc=qgis.utils.iface.mapCanvas() #llamada al canvas de Qgis (el mapa)
193             mc.setExtent(rect) #la vista de mapa será la extensión del rectánculo, es
                decir, el punto seleccionado
194             mc.refresh() #actualiza la vista de mapa
195             #seleccionar actual en el mapa
196             capas_seleccionadas=QgsProject.instance().mapLayersByName(self.txt_activa.text
                ())
197             capa_seleccionada=capas_seleccionadas[0]
198             clave=self.tw_seleccion.item(fila,8).text()
199             capa_seleccionada.selectByExpression("\"fid\"="+clave)
200             #foto
201             rutafoto=self.tw_seleccion.item(fila,7).text()
202             self.le_ruta.setText(rutafoto)
203             imagen=QPixmap(rutafoto)
204             self.lb_foto.setPixmap(imagen)
205             self.lb_foto.adjustSize()
206
207             #incidencia
208             inciden=self.tw_seleccion.item(fila,0).text()
209             if (inciden=="True"):
210                 self.cb_incidencia.setChecked(True)
211             else:
212                 self.cb_incidencia.setChecked(False)
213
214             #trazabilidad
215             trazab=self.tw_seleccion.item(fila,3).text()
216             self.te_traza.setText(trazab)
217
218             #descripcion
219             descrip=self.tw_seleccion.item(fila,2).text()
220             self.te_descripcion.setText(descrip)
221
222             #fid campo clave
223             fid=self.tw_seleccion.item(fila,8).text()
224             self.te_clave.setText(fid)
225
226             #fecha
227             fec=self.tw_seleccion.item(fila,1).text()
228             self.de_fecha.setDate(QDate.fromString(fec,"dd/MM/yyyy"))
229
230         def registra_cambios(self,event):
231             #    A C T U A L I Z A   L A   T A B L A   D E   A T R I B U T O S
232             #       S E G U N   S E   C A M B I A N   D A T O S   E N   F O R M U L A R I O
233
234             capas_seleccionadas=QgsProject.instance().mapLayersByName(self.txt_activa.text
                ())
235             capa_seleccionada=capas_seleccionadas[0]
236             fila=self.tw_seleccion.currentRow()
237             clave=int(self.tw_seleccion.item(fila,8).text())
238             descrip=self.te_descripcion.toPlainText()
239             #fec=self.de_fecha.date()
240             if(self.cb_incidencia.isChecked()):
241                 inciden=True
242             else:
243                 inciden=False
244
245             #empieza edicion
246             capa_seleccionada.startEditing()
247
248             #cambiamos todos los campos actualizando contenidos con la Qtable
249             capa_seleccionada.changeAttributeValue(clave, 13, descrip)
250             capa_seleccionada.changeAttributeValue(clave, 14, inciden)
251             #capa_seleccionada.changeAttributeValue(clave, 14, fec)
252
253             capa_seleccionada.commitChanges()
254             #cambia en la capa pero no en la tabla
255             #borramos filas de la tabla y volvemos a cargar:
256             self.tw_seleccion.setRowCount(0)
```

```
257             capas_seleccionadas=QgsProject.instance().mapLayersByName(self.txt_activa.text
                ())
258             capa_seleccionada=capas_seleccionadas[0]
259             registros=capa_seleccionada.getFeatures()
260             for registro in registros:
261                 self.tw_seleccion.insertRow(self.tw_seleccion.rowCount())
262                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,0,QtWidgets.
                    QTableWidgetItem(str(registro["incidencia"])))
263                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,1,QtWidgets.
                    QTableWidgetItem(registro["timestamp"].toString("dd/MM/yyyy")))
264                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,2,QtWidgets.
                    QTableWidgetItem(str(registro["DESCRIPCION"])))
265                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,3,QtWidgets.
                    QTableWidgetItem(str(registro["DESCRIPCION_TRAZA"])))
266                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,4,QtWidgets.
                    QTableWidgetItem(str(registro["CODIGO"])))
267                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,5,QtWidgets.
                    QTableWidgetItem(str(registro["longitude"])))
268                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,6,QtWidgets.
                    QTableWidgetItem(str(registro["latitude"])))
269                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,7,QtWidgets.
                    QTableWidgetItem(str(registro["photo"])))
270                 self.tw_seleccion.setItem(self.tw_seleccion.rowCount()-1,8,QtWidgets.
                    QTableWidgetItem(str(registro["fid"])))
271
```